Ollscoil na Gaillimhe
University of Galway

Autumn Examinations 2022-2023

Course Instance Code(s)	4BCT, 4BS
Exam(s)	B.Sc. (CS&IT) B.Sc.
Module Code(s)	CT404, CT336
Module(s)	Graphics and Image Processing
Paper No.	1

External Examiner(s)	Dr. R. Trestian
Internal Examiner(s)	Prof. M. Madden *Dr. S. Redfern

Instructions:

- Answer **three questions**
- Of which **at least one must be Graphics** (Q1, Q2, Q3)
- And **at least one must be Image Processing** (Q4, Q5)
- Your third question can come from either (Q1, Q2, Q3, Q4, Q5)
- All questions carry equal marks.

Duration	2 hours
No. of Pages	8
Discipline(s)	Computer Science
Course Co-ordinator(s)	Dr. C. O'Riordan

Requirements:

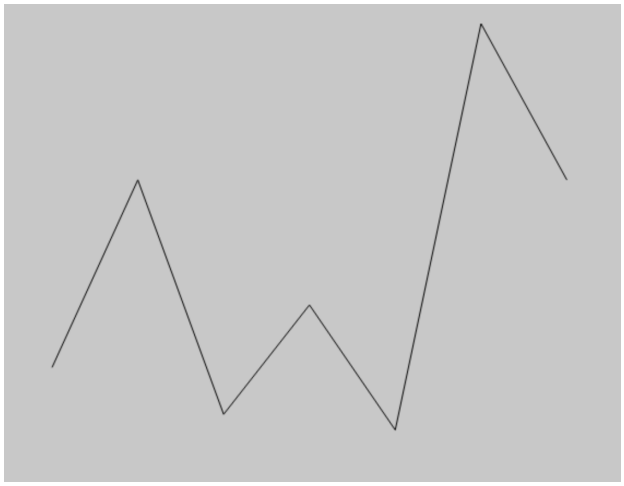
Release in Exam Venue	Yes [x]	No [ ]
MCQ Answersheet	Yes [ ]	No [x]
Handout	None	
Statistical/ Log Tables	None	
Cambridge Tables	None	
Graph Paper	None	
Log Graph Paper	None	
Graphic material in colour	Yes [x] No [ ]	

PTO

Q.1. (Graphics)

(i) Provide short sections of code illustrating **translation**, **rotation**, and **scaling** in either Canvas2D or Threejs. Note that the final page of this exam paper lists some commonly used functions in Canvas2D and Threejs. [10]

(ii) Using the code below as a starting point, write Javascript/Canvas2D code for use in the **draw()** function which will draw a line graph from the numbers contained in the **data[]** array. The graph should apply appropriate scales on the *x* and *y* axes, bearing in mind that the values in **data[]** may be changed, i.e. you should not hard-code the scales. There is no requirement to label the axes. [10]

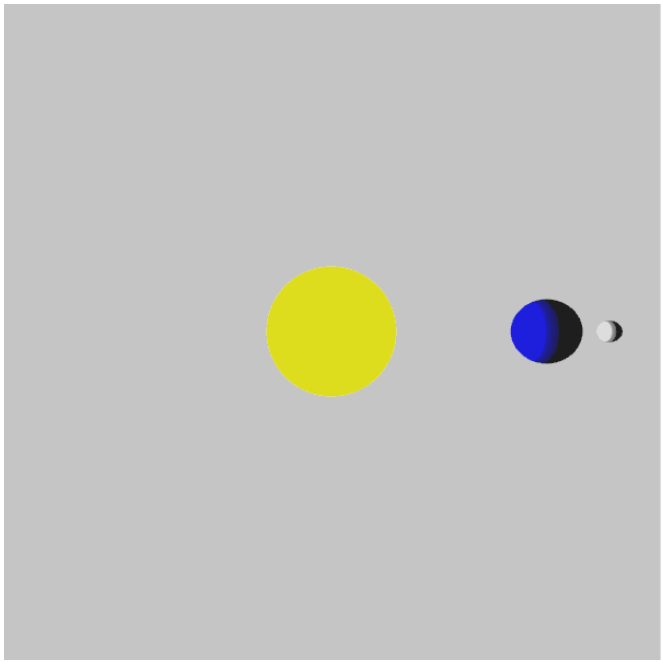


```
<html>
  <head>
    <script>
    function draw() {
        var canvas = document.getElementById("canvas");
        var ctx = canvas.getContext("2d");
        var data = [6, 18, 3, 10, 2, 28, 18];

        // to do: write code here to draw a line graph using Canvas2D

    }
    </script>
  </head>

<body onload='draw();'>
  <canvas id="canvas" width="600" height="450"></canvas>
</body>
</html>
```

PTO

Q.2. (Graphics)

Examine the **Javascript/Threejs** code provided below (and on the next page), which sets up a display of a sun, planet, and moon as in the picture below.



(i) In order for the program to provide an animation whereby the blue planet rotates around the yellow sun, while at the same time the grey moon rotates around the blue planet, it will be necessary to **nest their coordinate systems using pivots**. Explain what this means and why it is necessary. [7]

(ii) Provide the code changes that are needed in the **draw( )** function to establish this nesting. *Note that the final page of this exam paper lists some commonly used functions in Threejs.* [7]

(iii) Write suitable code for the **animate( )** function to apply a small amount of rotation per frame, to the planet pivot so that the planet orbits the sun, and to the moon pivot so that the moon orbits the planet. [6]

```
<html>
 <head>

  <script src="three.js"></script>
  <script>
   'use strict'

   var renderer, scene, camera;
   var sun, planet, moon;
```

PTO

```
function draw() {. // create renderer attached to HTML Canvas object
  var c = document.getElementById("canvas");
  renderer = new THREE.WebGLRenderer({ canvas: c, antialias: true });

  scene = new THREE.Scene();  // create the scenegraph
  scene.background = new THREE.Color( 0x333333 );

  var fov = 75;  // create a camera
  var aspect = 600/600;
  var near = 0.1;
  var far = 1000;
  camera = new THREE.PerspectiveCamera( fov, aspect, near, far );
  camera.position.set(0, 0, 20);
  camera.lookAt(new THREE.Vector3(0,0,0));

  var light = new THREE.PointLight(0xFFFFFF); // add a light to the scene (at location of the sun)
  light.position.set(0, 0, 0);
  scene.add(light);

  sun = new THREE.Mesh(. // create the sun, planet, and moon
    new THREE.SphereBufferGeometry(3,60,60),
    new THREE.MeshBasicMaterial({color: 0xffff00}) );
  sun.position.set(0, 0, 0);
  scene.add(sun);

  planet = new THREE.Mesh(
    new THREE.SphereBufferGeometry(1.5,60,60),
    new THREE.MeshLambertMaterial({color: 0x0000AA}) );
  planet.position.set(10, 0, 0);
  scene.add(planet);

  moon = new THREE.Mesh(
    new THREE.SphereBufferGeometry(0.5,60,60),
    new THREE.MeshLambertMaterial({color: 0x888888}) );
  moon.position.set(13, 0, 0);
  scene.add(moon);
  animate();
 }

 function animate() {
  setTimeout(animate, 20);
  renderer.render(scene, camera);
 }
</script>
</head>

<body onload="draw();">
 <canvas id="canvas" width="600" height="600"></canvas>
</body>
</html>
```

PTO

Q.3. (Graphics)

(i) Many of the techniques used in real-time 3D graphics programming attempt to maximise the realism of the rendered scene while processing a minimal number of polygons. With specific reference to the so-called **polygon budget**, and using diagrams where appropriate, **discuss** each of the following techniques. Your discussion should not only explain how they work, but also when they are appropriate, and their strengths and weaknesses (where appropriate). [12]

a) Bump Mapping
b) Back Face Culling
c) Billboards
d) Levels-of-Detail (LODs)

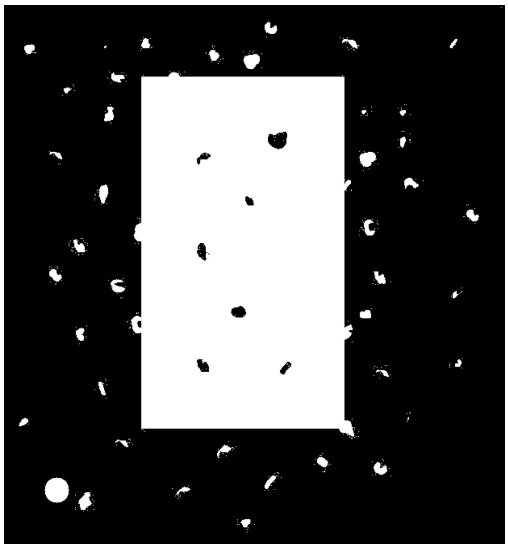
(ii) Real time graphics algorithms are categorised as operating in either **world space** or in **image space**. Explain the meaning of world space algorithms and image space algorithms from the point of view of the rendering pipeline. List one algorithm which could operate in world space (and explain why world space is appropriate to it) and list one algorithm which could operate in image space (and explain why image space is appropriate to it). [8]

PTO

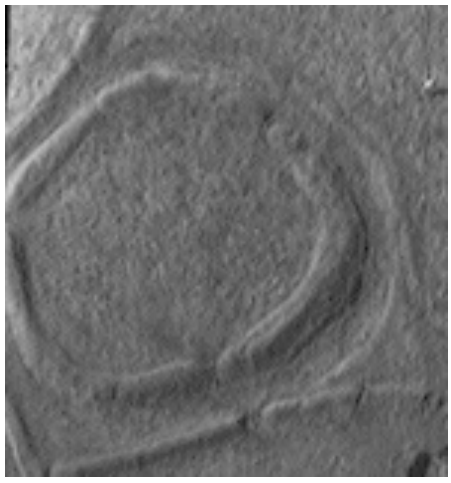
Q.4. (Image Processing)

(i) Describe the **mathematical morphology** approach to image processing. Outline some typical circumstances in which this approach is useful. [4]

(ii) The image below contains a large white rectangle and a number of patches of noise. Outline and defend a morphology-based algorithm for automatic isolation of the rectangle. Use sketches to indicate approximately how the image would appear following each step of your solution. [7]



(iii) Outline, at a high level, a suitable algorithm for extracting 3D information from a stereo pair of images, such as the image pair illustrated below. [9]

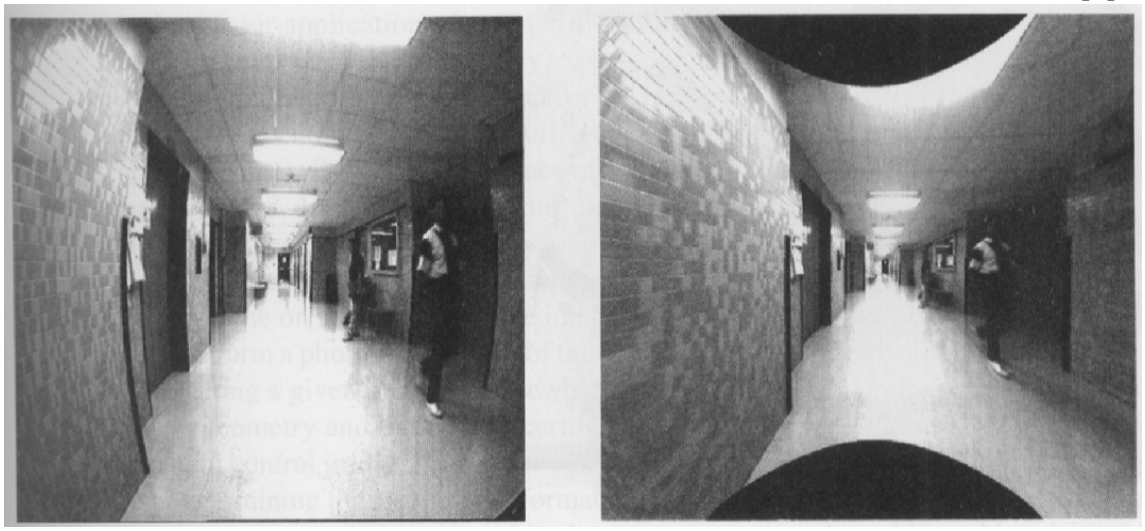


PTO

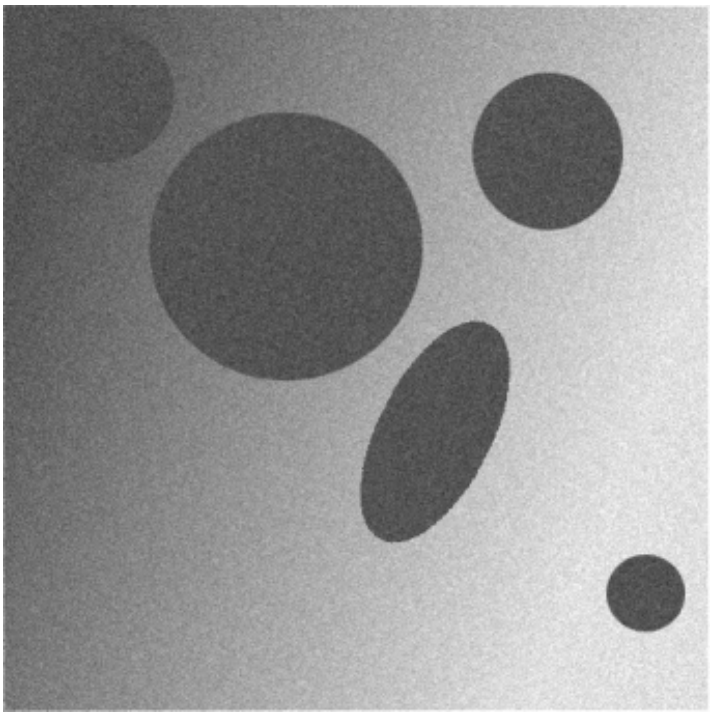
Q.5. (Image Processing)

(i) **Camera decalibration** is a technique for geometric correction of images which is often employed when sources of geometric error are poorly understood. With regard to camera decalibration:

- Outline the use of reference images such as grids of dots to construct and apply geometric corrections to images captured with a wide-angle lens (e.g. the image below). Use the terms 'control points', 'pixel filling', and 'bilinear interpolation' in your answer. [7]
- Explain why would you expect a reference image to be constructed with black markings on a white background (or white markings on a black background) [3]



(ii) In many cases, it may not be possible to robustly segment an image based on detected edges. One solution is to apply template matching approaches such as the **Hough Transform**. Explain in simple terms the operation of the Hough Transform as it applies to circles, indicating why it might be a good choice for automatic extraction of circle-like shapes such as those depicted in this noisy image. [10]



PTO

Some useful methods/properties of the Canvas 2D Context object:

Method/Property	Arguments/Values	Notes
fillRect	(Left, Top, Width, Height)	Draw a filled rectangle
beginPath	None	Start a stroked path
moveTo	(X, Y)	Move the graphics cursor
lineTo	(X, Y)	Draw a line from graphics cursor
stroke	None	End a stroked path
fillStyle	="rgb(R,G,B)"	Set fill colour
strokeStyle	="rgb(R,G,B)"	Set line colour
save	None	Save the current coordinate system
restore	None	Restore the last saved coord system
translate	(X,Y)	Translate the coordinate system
rotate	(angle)	Rotate the coordinate system clockwise, with angle in radians
scale	(X,Y)	Scale the coordinate system independently on the X and Y axes

Some useful objects/methods from the Threejs library:

Object/Method	Notes
new THREE.WebGLRenderer({canvas:c})	Constructs a renderer, attached to the Canvas object c
new THREE.PerspectiveCamera(fov,aspect,near,far)	Constructs a camera, with the specified field-of-view, aspect ratio, near clipping distance, far clipping distance
new THREE.Scene();	Constructs a scene
new THREE.PointLight(0xffffff);	Constructs a white point light
object.position.set(x,y,z)	Sets an object's x,y,z position relative to its parent
object.rotation.set(x,y,z)	Sets an object's x,y,z rotation (using Euler angles) relative to its parent
object.rotateOnAxis(new THREE.Vector3(0,1,0), 0.1)	Rotates object by 0.1 radians on the y axis
object1.add(object2)	Sets object2 as a child of object1
object.parent	Obtains a reference to the parent of object
camera.lookAt(new THREE.Vector3(0,0,0));	Turns a camera object to face the world coordinate 0,0,0
new THREE.BoxGeometry(20, 20, 20)	Constructs Box geometry, with specified width, height, depth
new THREE.MeshLambertMaterial({color: 0xfd59d7})	Constructs a Lambert (Phong) material of the specified colour
new THREE.Mesh(geometry, material)	Constructs a mesh using the specified geometry and material
renderer.render(scene, camera)	Uses a renderer to draw a scene as seen by a camera, onto the renderer's Canvas

END